Image Resolution Manipulation

A Project by

Juan Paolo G. Fernando
Ryan Jay P. Ferrera
Francis Jerome G. Tiausas

Submitted to

Luisito L. Agustin
Instructor, ELC 152

In Partial Fulfillment of the Requirements for the Course
ELC 152: Signal Processing

Department of Electronics, Computer and Communications Engineering
School of Science and Engineering
Loyola Schools
Ateneo de Manila University
Quezon City, Philippines

October 2010

Abstract

The higher the resolution of an image, the more detail it has. A single pixel is created by combining different values of the three color channels in the RGB color model, with each different combination resulting in a specific pixel color. Implemented with the use of wxDevC++, this project can open an image file of 8-bit resolution, change its resolution by altering the bit count for the three colors (RGB), and produce the edited image. The *Bitmap* and *Image* classes were used in order to design and implement the resolution functions, and a graphical user interface which illustrates the difference between the original and the processed image is also supported by the program.

Table of Contents

1. Project Overview

1.1. Objectives

The main objective of this project is to be able to produce a program that is able to lower the resolution of an image loaded into the program by altering the RGB color of the image.

Knowledge of C++ should first be guaranteed by the group since it was decided that this programming language will be used in creating the program. Compilation of the program will be used using Dev C++. wxDev C++, in particular, will be used for the creation of the graphical user interface (GUI) of the program, as it is a visual RAD designer for creating GUIs to cross-platform applications.

The program to be created should be able to open or load a digital image and show the original image in a window that has the ability of scrolling. The image can then be altered by allowing the user to change the RGB color model individually, and the edited image will be shown on a separate window that also has the ability of scrolling. In altering the colors, the user could input the desired number for the color or change it with increments of 1 using the arrows. Should the user input a number higher than 8, it will just reset itself back to 8. The user has the option to save the modified image in Bitmap or JPEG format.

1.2. Significance of the Project

There is still a considerable amount of people who have little to almost zero knowledge of image editing even though technology has taken great leaps in terms of advancements. To keep up with the fast pace of technology, programs should be more user-friendly for people to easily learn them. As technology advances, such as by coming up with better features which, more often than not, result to higher virtual size, programs

should be developed to be able to cope with such demands yet they are simple to use and yield results at a fast pace.

Digital images of today have become packed with more details of information to increase the quality of the digital image. The program was made with the increase of quality of digital image in mind as a factor. The significance of this project is to allow users a view of how a high quality image would look like should the resolution of that image be lowered in the aspects of its red, blue, and green, giving a glimpse of how image quality is compromised.

1.3. Scope and Limitations

The program is expected to be able to alter the resolution of an image by manipulating individually the values of R, G, and B. The values of R, G, and B are only from 0 to 255, represented as 8-bit data. Should the user input a value higher than 8, the edited image would return to it's original value for that specific color.

The program supports digital image formats of JPEG, BMP, GIF, and PNG. However, the manipulated image can only be saved in BMP or JPEG format.

1.4. Implementation Summary

Image classes, such as that of wxImage and wxBitmap, were used in implementing the Image Resolution Manipulation program. Other functions were used as well in the creation of the program. All of these were then used to construct the Graphical User interface.

Once an image is opened or loaded, the image is portrayed on a window with scrolling ability. This window is displayed on the left side of the main window. The user is then allowed to alter the R, G, and B of the image, effectively changing the resolution

of the original image. The edited image is then portrayed on a window with scrolling ability located on the right of the main window.

The program uses an algorithm that allows the user to decrease an image resolution by individually altering the values of R, G, and B. Files of conventional formats that is supported by the image class, such as JPEG and BMP, may be opened.

1.5 Important Terms

1.5.1. Digital Images

A digital image usually refers to a raster image, which is basically a data structure representing a rectangular grid of pixels. This raster image has a finite set of digital values, commonly referred to as pixels. Digital images are created by digitization, a process which involves transforming analog media into electronic data.

1.5.2. RGB

The RGB color model works by adding red, green, and blue light together in many different ways in order to produce a range of many colors and color variants. This color model is used in order to sense, represent, and display images in televisions, computers, other electronic systems, as well as in conventional photography. The intensity values of each of the three color channels range from 0 to 255, and it is by lowering the values for any of the three colors that a pixel of a specific color is created.

1.5.3. Image Resolution

The resolution of a digital image refers to the detail, or pixel count, that it possesses. An image possesses more detail as the resolution increases, and lowering the resolution of an image would result to an image of lesser quality. Since the RGB can be altered individually, the resulting image will show how much the colors are affected. For example, lowering the Red will result to an image that is more blue and more green.

1.5.4. Resolution of Digital Images

There are different ways as to how a resolution of a digital image can be described. Some of the known ways are namely through Spatial Resolution, Spectral Resolution, Temporal Resolution, Radiometric Resolution, and Pixel Resolution.

1.5.4.1. Spatial Resolution

One of the ways by which image resolution could be described is through Spatial Resolution. It is the measure of how closely lines can be resolved in an image, and is the number of independent pixel values in the image per unit length. It depends not only on the pixel resolution in pixels per inch, but also on the properties of the system that are responsible in creating the image.

1.5.4.2. Spectral Resolution

Spectral Resolution, or the resolving power of a spectrograph, is yet another way of how the resolution of a digital image can be described. For Digital color images or digital images that contain color information for each pixel, light of various ranges are characterized. Images that contain multiple ranges of light *resolve* even more detailed differences of wavelength than is needed to reproduce color, which would then result to the image having a higher spectral resolution.

1.5.4.3. Temporal Resolution

Temporal Resolution is defined as the measurement's accuracy with respect to time. There is a compromise between a measurement's temporal resolution and its spatial resolution. A lower temporal resolution is achieved as the distance increases. The reason for such a compromise is that it takes for photons with information to reach the observer, and that a change that the system might have occurred during this time.

While movies usually have a temporal resolution of 15 to 30 frame/s (frames per second), high-speed cameras can reach up to 100 to 1000 frame/s.

1.5.4.4. Radiometric Resolution

A system that can denote or characterize differences of intensity is determined by Radiometric Resolution. The differences are normally expressed as number of levels or number of bits, such as that of typical digital image file which consists of 8 bit or 256 levels. In theory, as the Radiometric Resolution becomes higher, representation of the differences of intensity becomes better and more subtle. In practice, however, noise level, rather than number of bits representation, limits the effective radiometric resolution.

1.5.4.5. Pixel Resolution

The last way to describe the resolution of a digital image is by Pixel Resolution. Even when international standards specify that it should not be used as such, often times in digital imaging pixel count refers the term resolution. The convention of describing pixel resolution is a set of two positive integers where the first number denotes the pixel columns, or width, and pixel rows, or height, such as that of 1024 by 720. Yet another convention that circulates is to say that resolution refers to the number of pixels in a digital image. To calculate for this, the pixel column is multiplied to the pixel row and then divided by 1 million, which would then result to a number in the megapixels range. Other popular conventions include referring to pixels per length or area unit, for example pixels per inch or pixels per square inch. Even though the aforementioned definitions pixel resolutions not are true resolution, they are popularly described as such, and also serve as the upper limit of image resolution.

When talking about the resolution of a digital image, pixel count is not a real measure of resolution simply because *color image sensors are set up to alternate color filter types over light sensitive individual pixel sensors.* Sensors cannot supply red, green, and blue for each pixel to be shown at full pixel count, which is a requirement for digital

images. In a full image sensor, pieces of information of red, green, and blue are the only ones that can be supplied by individual pixel element in a full image sensor. To be able to produce a unique set of three colors for each output pixel, an image has to be "downsampled" which will reduce its size, resulting to a reduction of visual image quality.

2. Important Classes

2.1. The wxImage Class

The wxImage class of wxDev C++ was used in the program. This class was used to be able to extract and manipulate image data on the per pixel level.

2.1.1. Functions of the wxImage class

Images can be created from data using either the wxImage() constructor or the ConvertToImage() function of the wxBitmap class. WxImage naturally supports only the Bitmap (.bmp) image format; however, image format handlers can also be manipulated to support other formats of images.

Pixel data is readily obtained from a wxImage object through the GetRed(), Get Green(), and GetBlue() functions. Alternatively, manipulation of this data is done using the SetRed(), SetGreen(), and SetBlue() functions. These functions are used to set the integer numerical value (0-255) of each color for each pixel. It is also worth noting that some image file formats may also support the GetAlpha() and SetAlpha() functions which allow access to the *"alpha"* or transparency component of the image.

The wxImage class also supports the GetHeight() and GetWidth() functions for obtaining the height and width of the image data.

2.2. The wxBitmap Class

The wxBitmap class was used to be able to open image files of any format, and then load them into the program's GUI to be displayed. Alternatively, they can be converted into wxImage objects to allow for direct manipulation of the contained pixel data.

2.2.1. Functions of the wxBitmap class

A wxBitmap object is typically created using either the wxBitmap() constructor or the LoadFile() function. Both functions take a wxString of the file path of the image to be loaded as an argument. The wxBitmap class contains the ConvertToImage() function which returns a wxImage object given a previously instantiated wxBitmap object.

The GetHeight() function of the class may be used to gain access to the height of the image in pixels, while the GetWidth() function, on the other hand, may be used to gain access to the width of the image in pixels.

2.3. The wxStaticBitmap class

The wxStaticBitmap class acts as the placeholder where the converted image data is displayed. Only the wxStaticBitmap constructor was used since that alone already provided the ability to display the image.

3. Algorithms

3.1. Image Resolution Change algorithm.

The algorithm used for changing the bit-resolution of the individual R, G, and B components of images is simply a bit-truncation function which drops the less significant bits of the color component data but retains the more significant ones depending on the target bit-resolution threshold. The resulting image would therefore have color component data comprising of fewer bits than that of the original.

The algorithm itself was implemented in the program by using C++'s native bit-shift operators, “<<” and “>>”, on the color component data. The program function, ResolutionChange(), is dedicated solely for this purpose. First, the individual R, G, and B color component data for each pixel of the image is extracted. The extracted color component data – originally composed of 8-bits each – is then shifted to the right by several units, discarding most of the less significant bits.

Red	$[1\,0\,1\,1\,0\,1\,1\,0]$
…	
Red >> 5	$[1\,0\,1\,1\,0\,1\,1\,0] \gg 5 = [0\,0\,0\,0\,0\,1\,0\,1]$

Thus, in order to represent this modified data in one of the conventional 8-bit image formats, the program would merely have to shift back the image as many units to the left as it was initially shifted to the right. This would place the more significant bits of the color component data back to where they originally should have been. The only difference this time is that the less significant bits of the data have all been replaced with zeroes.

Red >> 5	$[0\,0\,0\,0\,0\,1\,0\,1]$
…	
Red << 5	$[0\,0\,0\,0\,0\,1\,0\,1] \ll 5 = [1\,0\,1\,0\,0\,0\,0\,0]$

3.2. Euclidean Distance algorithm

Another important algorithm used in the program is the Euclidean Distance (ED) algorithm. This algorithm is used to generate pixel distance metrics based on the

individual color components of the resulting image against that of the original image. The equation for obtaining the Euclidean distance between two values is given as:

$$X = x_2 - x_1 \ ;$$

$$ED = \sum_{i=0}^{m} \sqrt{(X_i)^2}$$

In the context of this program, the Euclidean distance is computed between each color component of each pixel in the original image against that of each corresponding pixel in the modified image. The corresponding formula would then be:

$$ED = \sum_{i=0}^{m} \sum_{j=0}^{n} \sqrt{(R_{i,j})^2 + (G_{i,j})^2 + (B_{i,j})^2}$$

where $R_{i,j} = R_{original} - R_{modified}$, $G_{i,j} = G_{original} - G_{modified}$, $B_{i,j} = B_{original} - B_{modified}$,

n = image width,

m = image height;

The aforementioned formula is implemented in the PixelDistanceMetric() function of the program, where it is used to compute for the average Euclidean distance as well as the maximum Euclidean distance between the images.

4. Graphical User Interface

The graphical user interface for the program was designed specifically for the purpose of being able to compare an original, unmodified image with that of its "filtered" or modified counterpart. The main features of the GUI are the side-by-side comparison windows showing the original image on one and the "filtered" image on the other, an image metric information display at the top-right corner of the window, and the "filter" controls panel at the top-left corner.

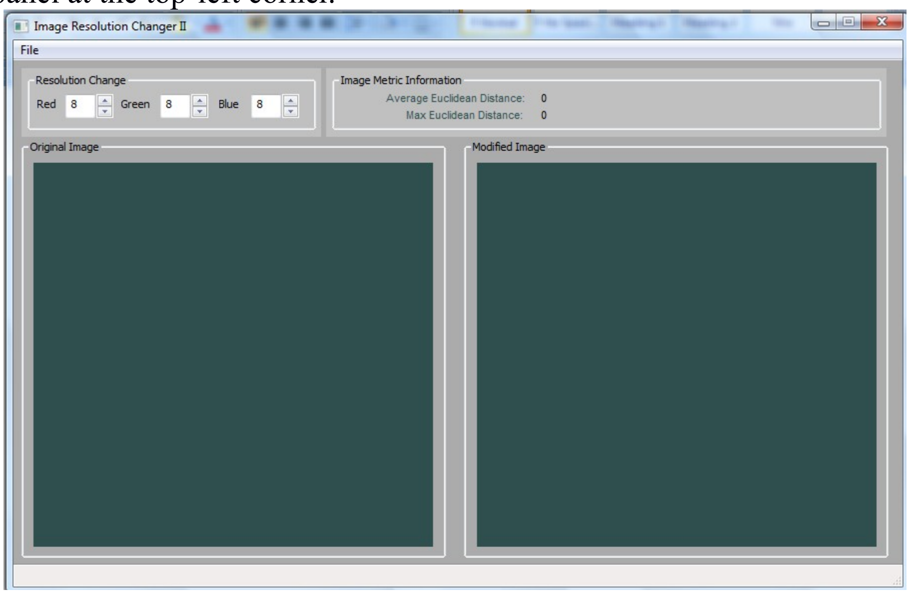

Fig. 1. The Graphical User Interface.

Fig. 2. The Graphical User Interface showing the original and manipulated

image with a combination of 3-3-3 for its RGB value.

In the context of this particular program, the “filter” controls are specifically geared towards being able to change the bit-resolution for the relevant color components of the chosen image. Thus, this panel contains the “spinner controls” for the changing the bit-resolutions of the red, green, and blue components of the image.

The program allows the user to save the modified image either as a Bitmap file or as a JPEG file.

As a special feature, whenever the user changes the resolution of the image, the coordinates of the original image are retained in the edited image. Whenever the original image is scrolled to a different position, the edited image follows even after the resolution has been changed.

Even though the program is specifically geared for the purpose of changing image bit-resolution, the GUI is flexible and adaptable enough to allow for the implementation of different filter algorithms using it as a platform.

5. Conclusions and Recommendations

5.1. Conclusions

The group was able to create a program that can change the bit-resolution of the individual color components of a given image, and display the original and resulting images side-by-side to the user via a GUI window. In addition to this, the program is also able to display the average Euclidean pixel color distance, as well as the maximum Euclidean pixel color distance. The program is able to work with a wide variety of input file types such as BMP, JPEG, GIF, and PNG.

5.2. Recommendations

The algorithm used for changing the bit-resolution in this program is a relatively low-level one, seeing as it works by merely truncating the color component data. Thus, the group recommends that future developments be geared towards the elaboration of the bit-resolution change technique, or perhaps towards formulating a newer and more innovative resolution change method. Implementation of image data recovery after moving to a lower bit-resolution should also be a focus for future development.

Future developments of this program could also include implementation of another algorithm for image resolution manipulation. The GUI would then differ by allowing the user to choose which combinations of images will be viewed side-by-side. This can be done by implementing a drop-down list where the original image, the modified image using this program's algorithm, and the modified image that uses a different algorithm can be selected for both the left and right windows.

Creating a new and unique image file format to store the bit-resolution changed image would also be an interesting path to venture into since it would make reducing the size of images via lowering the bit-resolution possible.

The program's basic framework is rather flexible and adaptable, and, thus, could readily be used for the implementation of other "filter" algorithms aside from just bit-resolution change.

Appendix 1
User's Manual

A1.1. Software Overview

A1.1.1. Minimum System Requirements

- For binary/executable
    - Windows 98/XP/Vista/Windows 7
    - 256 MB RAM
    - 30 MB free hard disk space (4MB for executable, extra for image)
    - 256 or more colors compatible display

A1.1.2. Features:

The *Image Resolution* allows the user to:

- Load Image files of any format;
- View the image file;
- View the original image along with the modified image;
- Alter the 8-bit RGB resolution of the image to any desired combination;
- View the modified image as it is being modified;
    - Saving the image as either BMP or JPEG.

A1.2. Availability

The *Image Resolution* is available through request from the authors.

A1.3. User's Guide

A1.3.1. Using the Software

The *Image Resolution* has a graphical user interface designed to run in Microsoft Windows. To run the program, open the “Image Resizer.exe” from the directory “<CD root directory>\software”.

A1.3.2. Loading the Image file

To load an image file, click on the “File” option on the Menu bar. Select “Load Image File” from the drop down menu that appears. This will launch a windows file dialog prompting the user to select an image file of any format from the hard disk. Upon selection of the Image file to be loaded, press ENTER or click OPEN. To cancel loading a file, press the CANCEL button or ESC key on the keyboard. The image shall be automatically displayed on both the “Original Image” and “Modified Image” boxes.

A1.3.3. Adjusting the 8-bit RGB Values of the Image

Two identical images are displayed at this point, with the Original Image displayed on the left side and the image to be modified on the right side. The resolution for the images at this point has not been manipulated. The user can manipulate the resolution of the image separately for each color. Upward and Downward buttons are present on the interface to raise and lower respectively the bit resolution of the image. By clicking the Upward and Downward buttons, the modified image instantly displays the changes made to the resolution of the image. The user can also write on the text box the number of bits for each color, ideally from 0-8. Also displayed on the User Interface is the Image Metric Information which displays the Average Euclidean Distance and the Max Euclidian Distance. These values show the difference of the modified image and the original image. This difference is measured using the Euclidean distance formula implemented as an algorithm within the program itself.

A1.3.4. Saving the modified image

The user can opt to save the manipulated image for later viewing. The saved image can either be Bitmap (.bmp) or JPEG (.jpg).

A1.3.5. Exiting the program

The user can choose between two ways on exiting the program. You can click on the X button on the title bar or select File from the Menu bar and then selecting Exit from the drop-down menu.

Appendix 2. Source Code

A2.1 Source Files

A2.1.1. image_resolution_changeApp.h

```
//-----------------------------------------------------------------------
------
//
// Name:         image_resolution_changeApp.h
// Author:       user
// Created:      9/19/2010 2:34:26 PM
// Description:
//
//-----------------------------------------------------------------------
------

#ifndef __IMAGE_RESOLUTION_CHANGEFRMApp_h__
#define __IMAGE_RESOLUTION_CHANGEFRMApp_h__

#ifdef __BORLANDC__
      #pragma hdrstop
#endif

#ifndef WX_PRECOMP
      #include <wx/wx.h>
#else
      #include <wx/wxprec.h>
#endif

class image_resolution_changeFrmApp : public wxApp
{
      public:
            bool OnInit();
            int OnExit();
};
```

```
#endif
```

A2.1.2. image_resolution_changeApp.cpp

```
//---------------------------------------------------------------------------
------
//
// Name:        image_resolution_changeApp.cpp
// Author:      user
// Created:     9/19/2010 2:34:26 PM
// Description:
//
//---------------------------------------------------------------------------
------

#include "image_resolution_changeApp.h"
#include "image_resolution_changeFrm.h"

IMPLEMENT_APP(image_resolution_changeFrmApp)

bool image_resolution_changeFrmApp::OnInit()
{
    wxInitAllImageHandlers();
                    image_resolution_changeFrm*     frame     =     new
image_resolution_changeFrm(NULL);
    SetTopWindow(frame);
    frame->Show();
    return true;
}

int image_resolution_changeFrmApp::OnExit()
{
      return 0;
}
```

A.2.1.3. image_resolution_changeFrm.h

```
///------------------------------------------------------------------
```

```
///
/// @file       image_resolution_changeFrm.h
/// @author     user
/// Created:    9/19/2010 2:34:27 PM
/// @section    DESCRIPTION
///             image_resolution_changeFrm class declaration
///
///------------------------------------------------------------------

#ifndef __IMAGE_RESOLUTION_CHANGEFRM_H__
#define __IMAGE_RESOLUTION_CHANGEFRM_H__

#ifdef __BORLANDC__
      #pragma hdrstop
#endif

#ifndef WX_PRECOMP
      #include <wx/wx.h>
      #include <wx/frame.h>
#else
      #include <wx/wxprec.h>
#endif

//Do not add custom headers between
//Header Include Start and Header Include End.
//wxDev-C++ designer will remove them. Add custom headers after the
block.
////Header Include Start
#include <wx/menu.h>
#include <wx/filedlg.h>
#include <wx/scrolwin.h>
#include <wx/statusbr.h>
#include <wx/stattext.h>
#include <wx/spinctrl.h>
#include <wx/statbox.h>
#include <wx/panel.h>
```

```
////Header Include End
#include <wx/statbmp.h>

////Dialog Style Start
#undef image_resolution_changeFrm_STYLE
#define image_resolution_changeFrm_STYLE wxCAPTION | wxSYSTEM_MENU |
wxMINIMIZE_BOX | wxCLOSE_BOX
////Dialog Style End

class image_resolution_changeFrm : public wxFrame
{
	private:
		DECLARE_EVENT_TABLE();

	public:
		image_resolution_changeFrm(wxWindow *parent, wxWindowID id =
1, const wxString &title = wxT("image_resolution_change"), const
wxPoint& pos = wxDefaultPosition, const wxSize& size = wxDefaultSize,
long style = image_resolution_changeFrm_STYLE);
		virtual ~image_resolution_changeFrm();
		void WxSpinCtrl1Updated(wxSpinEvent& event );
		void image_resolution_changeFrmActivate(wxActivateEvent&
event);
		void Mnuloadimagefile1034Click(wxCommandEvent& event);
		void SaveModifiedImage(wxCommandEvent& event);
		void LoadImage(wxCommandEvent& event);
		void ExitProgram(wxCommandEvent& event);
		void SaveModifiedImageAsJpeg(wxCommandEvent& event);

	private:
		//Do not add custom control declarations between
		//GUI Control Declaration Start and GUI Control Declaration
End.
		//wxDev-C++ will remove them. Add custom code after the
block.
		////GUI Control Declaration Start
```

```
            wxFileDialog *WxSaveFileDialog1;
            wxMenuBar *WxMenuBar1;
            wxFileDialog *WxOpenFileDialog1;
            wxScrolledWindow *WxScrolledWindow2;
            wxScrolledWindow *WxScrolledWindow1;
            wxStaticBox *WxStaticBox4;
            wxStaticBox *WxStaticBox3;
            wxStatusBar *WxStatusBar1;
            wxStaticText *WxStaticText7;
            wxStaticText *WxStaticText6;
            wxStaticText *WxStaticText5;
            wxStaticText *WxStaticText4;
            wxStaticBox *WxStaticBox2;
            wxPanel *WxPanel2;
            wxStaticText *WxStaticText3;
            wxStaticText *WxStaticText2;
            wxStaticText *WxStaticText1;
            wxSpinCtrl *WxSpinCtrl3;
            wxSpinCtrl *WxSpinCtrl2;
            wxSpinCtrl *WxSpinCtrl1;
            wxStaticBox *WxStaticBox1;
            wxPanel *WxPanel1;
            ////GUI Control Declaration End
            wxStaticBitmap *OriginalBitmap;
            wxStaticBitmap *ModifiedBitmap;
            void ResolutionChange(wxImage& my_image);
            void    PixelDistanceMetric(wxImage&    original,    wxImage&
modified);
      private:
            //Note: if you receive any error with these enum IDs, then
you need to
            //change your old form code that are based on the #define
control IDs.
            //#defines may replace a numeric value for the enum names.
            //Try copy and pasting the below block in your old form
header files.
```

```
        enum
        {
            ////GUI Enum Control ID Start
            ID_MNU_FILE_1016 = 1016,
            ID_MNU_LOADIMAGEFILE_1034 = 1034,
            ID_MNU_SAVEMODIFIEDIMAGE_1035 = 1035,
            ID_MNU_SAVEMODIFIEDIMAGEASJPEG_1037 = 1037,
            ID_MNU_EXIT_1036 = 1036,

            ID_WXSCROLLEDWINDOW2 = 1029,
            ID_WXSCROLLEDWINDOW1 = 1028,
            ID_WXSTATICBOX4 = 1027,
            ID_WXSTATICBOX3 = 1026,
            ID_WXSTATUSBAR1 = 1012,
            ID_WXSTATICTEXT7 = 1033,
            ID_WXSTATICTEXT6 = 1032,
            ID_WXSTATICTEXT5 = 1031,
            ID_WXSTATICTEXT4 = 1030,
            ID_WXSTATICBOX2 = 1018,
            ID_WXPANEL2 = 1011,
            ID_WXSTATICTEXT3 = 1024,
            ID_WXSTATICTEXT2 = 1023,
            ID_WXSTATICTEXT1 = 1022,
            ID_WXSPINCTRL3 = 1021,
            ID_WXSPINCTRL2 = 1020,
            ID_WXSPINCTRL1 = 1019,
            ID_WXSTATICBOX1 = 1017,
            ID_WXPANEL1 = 1009,
            ////GUI Enum Control ID End
            ID_DUMMY_VALUE_ //don't remove this value unless you
have other enum values
        };

    private:
        void OnClose(wxCloseEvent& event);
        void CreateGUIControls();
```

```
};

#endif
```

A.2.1.4. image_resolution_changeFrm.cpp

```
///-----------------------------------------------------------------
///
/// @file      image_resolution_changeFrm.cpp
/// @author    user
/// Created:   9/19/2010 2:34:27 PM
/// @section   DESCRIPTION
///            image_resolution_changeFrm class implementation
///
///------------------------------------------------------------------

#include "image_resolution_changeFrm.h"

//Do not add custom headers between
//Header Include Start and Header Include End
//wxDev-C++ designer will remove them
////Header Include Start
////Header Include End

//--------------------------------------------------------------------
-------
// image_resolution_changeFrm
//--------------------------------------------------------------------
-------
//Add Custom Events only in the appropriate block.
//Code added in other places will be removed by wxDev-C++
////Event Table Start
BEGIN_EVENT_TABLE(image_resolution_changeFrm,wxFrame)
	////Manual Code Start
	////Manual Code End

	EVT_CLOSE(image_resolution_changeFrm::OnClose)
```

```
EVT_ACTIVATE(image_resolution_changeFrm::image_resolution_changeFrmActi
vate)
	EVT_MENU(ID_MNU_LOADIMAGEFILE_1034,
image_resolution_changeFrm::LoadImage)
	EVT_MENU(ID_MNU_SAVEMODIFIEDIMAGE_1035,
image_resolution_changeFrm::SaveModifiedImage)
	EVT_MENU(ID_MNU_SAVEMODIFIEDIMAGEASJPEG_1037,
image_resolution_changeFrm::SaveModifiedImageAsJpeg)
	EVT_MENU(ID_MNU_EXIT_1036,
image_resolution_changeFrm::ExitProgram)

EVT_SPINCTRL(ID_WXSPINCTRL3,image_resolution_changeFrm::WxSpinCtrl1Upda
ted)

EVT_SPINCTRL(ID_WXSPINCTRL2,image_resolution_changeFrm::WxSpinCtrl1Upda
ted)

EVT_SPINCTRL(ID_WXSPINCTRL1,image_resolution_changeFrm::WxSpinCtrl1Upda
ted)
END_EVENT_TABLE()
////Event Table End

image_resolution_changeFrm::image_resolution_changeFrm(wxWindow
*parent, wxWindowID id, const wxString &title, const wxPoint &position,
const wxSize& size, long style)
: wxFrame(parent, id, title, position, size, style)
{
	CreateGUIControls();
}

image_resolution_changeFrm::~image_resolution_changeFrm()
{
}

void image_resolution_changeFrm::CreateGUIControls()
```

```
{
	//Do not add custom code between
	//GUI Items Creation Start and GUI Items Creation End
	//wxDev-C++ designer will remove them.
	//Add the custom code before or after the blocks
	////GUI Items Creation Start

	WxPanel1 = new wxPanel(this, ID_WXPANEL1, wxPoint(8, 8), wxSize(300, 69));
	WxPanel1->SetBackgroundColour(wxColour(wxT("LIGHT GREY")));

	WxStaticBox1 = new wxStaticBox(WxPanel1, ID_WXSTATICBOX1, wxT("Resolution Change"), wxPoint(6, 5), wxSize(289, 57));

	WxSpinCtrl1 = new wxSpinCtrl(WxPanel1, ID_WXSPINCTRL1, wxT("8"), wxPoint(44, 26), wxSize(48, 24), wxSP_ARROW_KEYS, 0, 8, 8);

	WxSpinCtrl2 = new wxSpinCtrl(WxPanel1, ID_WXSPINCTRL2, wxT("8"), wxPoint(139, 26), wxSize(48, 24), wxSP_ARROW_KEYS, 0, 8, 8);

	WxSpinCtrl3 = new wxSpinCtrl(WxPanel1, ID_WXSPINCTRL3, wxT("8"), wxPoint(230, 26), wxSize(48, 24), wxSP_ARROW_KEYS, 0, 8, 8);

	WxStaticText1 = new wxStaticText(WxPanel1, ID_WXSTATICTEXT1, wxT("Red"), wxPoint(16, 29), wxDefaultSize, 0, wxT("WxStaticText1"));

	WxStaticText2 = new wxStaticText(WxPanel1, ID_WXSTATICTEXT2, wxT("Green"), wxPoint(100, 29), wxDefaultSize, 0, wxT("WxStaticText2"));

	WxStaticText3 = new wxStaticText(WxPanel1, ID_WXSTATICTEXT3, wxT("Blue"), wxPoint(198, 29), wxDefaultSize, 0, wxT("WxStaticText3"));

	WxPanel2 = new wxPanel(this, ID_WXPANEL2, wxPoint(313, 8), wxSize(561, 69));
	WxPanel2->SetBackgroundColour(wxColour(wxT("LIGHT GREY")));
```

```
WxStaticBox2 = new wxStaticBox(WxPanel2, ID_WXSTATICBOX2, wxT("Image Metric Information"), wxPoint(6, 5), wxSize(550, 57));

WxStaticText4 = new wxStaticText(WxPanel2, ID_WXSTATICTEXT4, wxT("Average Euclidean Distance:"), wxPoint(60, 23), wxDefaultSize, 0, wxT("WxStaticText4"));
WxStaticText4->SetForegroundColour(wxColour(wxT("DARK SLATE GREY")));
WxStaticText4->SetFont(wxFont(8, wxSWISS, wxNORMAL, wxNORMAL, false));

WxStaticText5 = new wxStaticText(WxPanel2, ID_WXSTATICTEXT5, wxT("Max Euclidean Distance:"), wxPoint(80, 40), wxDefaultSize, 0, wxT("WxStaticText5"));
WxStaticText5->SetForegroundColour(wxColour(wxT("DARK SLATE GREY")));
WxStaticText5->SetBackgroundColour(wxColour(wxT("LIGHT GREY")));
WxStaticText5->SetFont(wxFont(8, wxSWISS, wxNORMAL, wxNORMAL, false));

WxStaticText6 = new wxStaticText(WxPanel2, ID_WXSTATICTEXT6, wxT("0"), wxPoint(215, 23), wxDefaultSize, 0, wxT("WxStaticText6"));
WxStaticText6->SetFont(wxFont(8, wxSWISS, wxNORMAL, wxNORMAL, false));

WxStaticText7 = new wxStaticText(WxPanel2, ID_WXSTATICTEXT7, wxT("0"), wxPoint(215, 40), wxDefaultSize, 0, wxT("WxStaticText7"));
WxStaticText7->SetFont(wxFont(8, wxSWISS, wxNORMAL, wxNORMAL, false));

WxStatusBar1 = new wxStatusBar(this, ID_WXSTATUSBAR1);

WxStaticBox3 = new wxStaticBox(this, ID_WXSTATICBOX3, wxT("Original Image"), wxPoint(8, 80), wxSize(425, 420));
```

```
	WxStaticBox4 = new wxStaticBox(this, ID_WXSTATICBOX4, wxT("Modified Image"), wxPoint(451, 80), wxSize(425, 420));

	WxScrolledWindow1 = new wxScrolledWindow(this, ID_WXSCROLLEDWINDOW1, wxPoint(20, 103), wxSize(400, 386), wxVSCROLL | wxHSCROLL);
	WxScrolledWindow1->SetBackgroundColour(wxColour(wxT("DARK SLATE GREY")));

	WxScrolledWindow2 = new wxScrolledWindow(this, ID_WXSCROLLEDWINDOW2, wxPoint(464, 103), wxSize(400, 386), wxVSCROLL | wxHSCROLL);
	WxScrolledWindow2->SetBackgroundColour(wxColour(wxT("DARK SLATE GREY")));

	WxOpenFileDialog1 =  new wxFileDialog(this, wxT("Choose a file"), wxT(""), wxT(""), wxT("*.*"), wxOPEN);

	WxMenuBar1 = new wxMenuBar();
	wxMenu *ID_MNU_FILE_1016_Mnu_Obj = new wxMenu(0);
	ID_MNU_FILE_1016_Mnu_Obj->Append(ID_MNU_LOADIMAGEFILE_1034, wxT("Load Image File"), wxT(""), wxITEM_NORMAL);
	ID_MNU_FILE_1016_Mnu_Obj->Append(ID_MNU_SAVEMODIFIEDIMAGE_1035, wxT("Save Modified Image as Bitmap"), wxT(""), wxITEM_NORMAL);
	ID_MNU_FILE_1016_Mnu_Obj->Append(ID_MNU_SAVEMODIFIEDIMAGEASJPEG_1037, wxT("Save Modified image as Jpeg"), wxT(""), wxITEM_NORMAL);
	ID_MNU_FILE_1016_Mnu_Obj->Append(ID_MNU_EXIT_1036, wxT("Exit"), wxT(""), wxITEM_NORMAL);
	WxMenuBar1->Append(ID_MNU_FILE_1016_Mnu_Obj, wxT("File"));
	SetMenuBar(WxMenuBar1);

	WxSaveFileDialog1 =  new wxFileDialog(this, wxT("Choose a file"), wxT(""), wxT(""), wxT("*.*"), wxSAVE);

	SetStatusBar(WxStatusBar1);
```

```
	SetTitle(wxT("Image Resolution Changer II"));
	SetIcon(wxNullIcon);
	SetSize(8,8,896,575);
	Center();

	////GUI Items Creation End
	OriginalBitmap = new wxStaticBitmap(WxScrolledWindow1,-
1,wxNullBitmap);
	ModifiedBitmap = new wxStaticBitmap(WxScrolledWindow2,-
1,wxNullBitmap);
}

void image_resolution_changeFrm::OnClose(wxCloseEvent& event)
{
	Destroy();
}
/*
 * SaveModifiedImage
 */
void image_resolution_changeFrm::SaveModifiedImage(wxCommandEvent&
event)
{
	// insert your code here
	WxSaveFileDialog1->ShowModal();
	if (WxSaveFileDialog1->GetPath().IsEmpty()) return;
	wxImage save_temp = ModifiedBitmap->GetBitmap().ConvertToImage();
	save_temp.SaveFile(_T(WxSaveFileDialog1->GetPath()
+".bmp"),wxBITMAP_TYPE_BMP);
}
/*
 * Mnusavemodifiedimageasjpeg1037Click
 */
void
image_resolution_changeFrm::SaveModifiedImageAsJpeg(wxCommandEvent&
event)
{
```

```
      // insert your code here
      WxSaveFileDialog1->ShowModal();
      if (WxSaveFileDialog1->GetPath().IsEmpty()) return;
      wxImage save_temp = ModifiedBitmap->GetBitmap().ConvertToImage();
      save_temp.SaveFile(_T(WxSaveFileDialog1->GetPath()
+".jpeg"),wxBITMAP_TYPE_JPEG);
}
/*
 * LoadImage
 */
void image_resolution_changeFrm::LoadImage(wxCommandEvent& event)
{
      // insert your code here
      WxOpenFileDialog1->ShowModal();
    if (WxOpenFileDialog1->GetPath().IsEmpty()) return;
           wxBitmap   clone   =   wxBitmap(WxOpenFileDialog1->GetPath(),
wxBITMAP_TYPE_ANY);
    wxImage original = clone.ConvertToImage();
    wxImage clone_holder = clone.ConvertToImage();
    ResolutionChange(clone_holder);
    PixelDistanceMetric(original,clone_holder);
    OriginalBitmap = new wxStaticBitmap(WxScrolledWindow1, -1, clone );
         ModifiedBitmap   =   new   wxStaticBitmap(WxScrolledWindow2,   -1,
wxBitmap(clone_holder) );
                                                       WxScrolledWindow1-
>SetScrollbars(10,10,original.GetWidth()/10,original.GetHeight()/10);
    WxScrolledWindow1->Scroll(1,1);
                                                       WxScrolledWindow2-
>SetScrollbars(10,10,clone_holder.GetWidth()/10,clone_holder.GetHeight(
)/10);
    WxScrolledWindow2->Scroll(1,1);
    Refresh();
}

/*
 * ExitProgram
```

```
 */
void image_resolution_changeFrm::ExitProgram(wxCommandEvent& event)
{
	// insert your code here
	Destroy();
}
/*
 * image_resolution_changeFrmActivate
 */
void
image_resolution_changeFrm::image_resolution_changeFrmActivate(wxActiva
teEvent& event)
{
	// insert your code here
}

/*
 * WxSpinCtrl1Updated
 */
void image_resolution_changeFrm::WxSpinCtrl1Updated(wxSpinEvent& event )
{
	// insert your code here
	wxImage clone = OriginalBitmap->GetBitmap().ConvertToImage();
	wxImage original = OriginalBitmap->GetBitmap().ConvertToImage();
	ResolutionChange(clone);
	PixelDistanceMetric(original,clone);
	ModifiedBitmap   =   new   wxStaticBitmap(WxScrolledWindow2,   -1,
wxBitmap(clone) );
    ModifiedBitmap->Enable();
    int stx = 0, sty = 0;
    WxScrolledWindow1->GetViewStart(&stx,&sty);
                                                     WxScrolledWindow2-
>SetScrollbars(10,10,clone.GetWidth()/10,clone.GetHeight()/10);
    WxScrolledWindow2->Scroll(stx,sty);
    WxScrolledWindow2->Refresh();
}
```

```
/*
 * Resolution Change
 */
void image_resolution_changeFrm::ResolutionChange(wxImage& my_image)
{
    long chr = (long)WxSpinCtrl1->GetValue();
    long chg = (long)WxSpinCtrl2->GetValue();
    long chb = (long)WxSpinCtrl3->GetValue();
    chr = 8-chr;
    chg = 8-chg;
    chb = 8-chb;
    for (int i=0; i<my_image.GetWidth(); i++){
        for(int j=0; j<my_image.GetHeight(); j++){
            int tempr = my_image.GetRed(i,j);
            tempr >>= chr; tempr <<= chr;
            int tempg = my_image.GetGreen(i,j);
            tempg >>= chg; tempg <<= chg;
            int tempb = my_image.GetBlue(i,j);
            tempb >>= chb; tempb <<= chb;
            my_image.SetRGB(i,j,tempr,tempg,tempb);
        }
    }
}
/*
 * Pixel Distance Metric
 */
void image_resolution_changeFrm::PixelDistanceMetric(wxImage& original,
wxImage& modified)
{
    float sum = 0;
    float isum = 0;
    float max = 0;
    for (int i=0; i<original.GetWidth(); i++){
        for (int j=0; j<original.GetHeight(); j++){
            isum = 0;
            isum += pow(original.GetRed(i,j)-modified.GetRed(i,j),2);
```

```
                                isum   +=   pow(original.GetGreen(i,j)-
modified.GetGreen(i,j),2);
            isum += pow(original.GetBlue(i,j)-modified.GetBlue(i,j),2);
            sum += sqrt(isum);
            if ( isum > max ) max = sqrt(isum);
        }
    }
    int total = original.GetWidth()*original.GetHeight();
    sum /= total;
    wxString nWxString;
    nWxString << sum;
    WxStaticText6->SetLabel(nWxString);
    nWxString = "";
    nWxString << max;
    WxStaticText7->SetLabel(nWxString);
}
```

Bibliography

Plataniotis, Konstantinos, and Anastasios Venetsanopoulos. *Color Image Processing and Applications*. Germany: Springer-Verlag 2000.

Plastock, Roy, Zhigang Xiang. *Schaum's Outlines: Computer Graphics*. United States of America: McGraw-Hill 2000.

"Spectral resolution". Space Telescope Science Institute. 17 Sept 2010. <http://www.stsci.edu/hst/stis/performance/spectral_resolution/>

"Image Resolution". Digital Expert. 17 Sept 2010. <http://dx.sheridan.com/advisor/image_resolution.html >

"Temporal Resolution". Range View. 25 Nov 2002. 17 Sept 2010. <http://rangeview.arizona.edu/Glossary/tempres.html >

Roebling, Robert, Smart, Julian, et al. "wxBitmap." wxWidgets. Feb 2010. 17 Sept 2010. <http://docs.wxwidgets.org/2.8/wx_wxbitmap.html#wxbitmap>

Roebling, Robert, Smart, Julian, et al. "wxImage." wxWidgets. Feb 2010 17 Sept 2010. <http://docs.wxwidgets.org/2.8/wx_wximage.html#wximage>